Applications de la Programmation Linéaire et Semi-Définie Positive

Bertrand Estellon, Frédéric Havet et Yann Vaxès.

JCALM Mai 2009

Ces notes ont été rédigées à la suite de la journée CALM à Marseille le 07 mai 2009. Elles reprennent le contenu des exposés de V. Chepoi, J. Galtier, P. Bonami, B. Estellon et Y. Vaxès,

1 Coupe multiterminaux (Multiway cut)

Soit $G = (V,E)$ un graphe muni d'une valuation des arêtes $w : E \to \mathbb{R}^+$ et k sommets appelés *terminaux* $s_1, \ldots, s_k$.

Une *coupe* est une partition de l'ensemble de sommets de G. Une $(s_1, \ldots, s_k)$-*coupe* est une partition $\Pi = (V_1, \ldots, V_k)$ de V telle que $s_i \in V_i$ pour tout $1 \leq i \leq k$.

Soit $\Pi = (V_1, \ldots, V_k)$ une coupe. Une arête e est *transverse* si ses deux extrémités sont dans des parties différentes. Autrement dit, $e = uv$, $u \in V_i$, $v \in V_j$ et $i \neq j$. L'ensemble des arêtes transverses de la partition Π est noté $E(\Pi)$. Le *poids* d'une coupe Π est la somme des poids des arêtes transverses: $w(\Pi) = \sum_{e \in E(\Pi)} w(e)$.

Le problème de la coupe multiterminaux s'exprime de la façon suivante:

Problème 1 (Coupe Multiterminaux)
Instance: Un graphe $G = (V,E)$, une valuation des arêtes $w : E \to \mathbb{R}^+$ et k sommets $s_1, \ldots, s_k$.
Question: Trouver une $(s_1, \ldots, s_k)$-coupe de poids minimun.

Pour $k = 2$, c'est le problème classique de la (s,t)-coupe minimum qui est polynomial. C'est d'ailleurs un cas particulier de la programmation linéaire. Pour $k \geq 3$, le problème est NP-difficile [4].

1.1 Première approximation

Comme l'ont montré Dalhaus et al. [4], en utilisant la résolution de la coupe minimum et en l'appliquant k fois on peut obtenir un algorithme $2 - \frac{2}{k}$-approché pour le problme de la coupe multiterminaux.

Pour cela pour tout i, on prend le graphe G_i obtenu à partir de G en identifiant les s_j, $j \neq i$ en un sommet t_i. On résoud ensuite la coupe minimum dans G_i entre s_i et t_i. On obtient alors une coupe minimum C_i qui sépare S_i de t_i dans G_i et donc s_i de tous les autres s_j dans G. L'union des $k-1$ coupes C_i les plus légères on obtient une $(s_1, \ldots, s_k)$-coupe.

1.2 Relaxation en programmation linéaire

Soit Δ_k l'emsemble des vecteurs de $\mathbb{R}^k$ à coordonnées positives de somme 1. Formellement, $\Delta_k = \{x \mid \sum_{i=1}^k x_i = 1 \text{ et } x_i \geq 0, \forall 1 \leq i \leq k\}$.

Notons également e_i le vecteur de Δ_k dont toutes les coordonnées sont nulles sauf la $i^{\text{ème}}$ qui vaut 1.

Le problème de la coupe multiterminaux peut être relâché de la manière suivante:

Problème 2

$$\begin{array}{rc} \text{Minimiser} & \sum_{uv\in E} w(uv)d(uv) \\ \text{Sous les contraintes :} & d(uv) = \frac{1}{2}\sum_{i=1}^{k} |x_u^i - x_v^i| \qquad (\star) \\ & x_v \in \Delta_k \\ & x_{s_i} = e_i \end{array}$$

Ce probème est un programme linéaire. En effet, l'équation $(\star)$ peut être classiquement mise sous forme linéaire en la remplaçant par les contraintes suivantes:

$$d(uv) = \frac{1}{2}\sum x_{u,v}^i$$
$$x_{uv}^i \geq x_u^i - x_v^i \text{ pour tout } 1 \leq i \leq k$$
$$x_{uv}^i \geq x_v^i - x_u^i \text{ pour tout } 1 \leq i \leq k$$

Le problème de la coupe multiterminaux est équivalent au Problème 2 pour lequel la contrainte $x_v \in \Delta_k$ est remplacée par $x_v \in \{e_i \mid 1 \leq i \leq k\}$. En effet, la partition est alors donnée par $V_i = \{v \mid x_v = e_i\}$.

Lemme 3 *Si x est une solution admissible au Problème 2 alors quitte à modifier l'entrée (et donc x en conséquence), on peut supposer que pour toute arête uv x_u et x_v diffèrent d'au plus 2 coordonnées.*

Preuve. Si jamais x_u et x_v diffèrent par $m > 2$ coordonnées, on subdivise l'arête uv. Précisément, on remplace l'arête uv par deux arêtes uw et wv que l'on value par $w(uw) = w(vw) = w(uv)$. On prend alors pour x_w un vecteur qui diffèrent de deux coordonnées avec u et strictement moins de m coordonnées avec v. Notons qu'alors $d(uv) = d(uw) + d(wv)$. □

1.3 Algorithme d'approximation

Nous donnons maintenant un algorithme qui donne une solution approchée au problème de coupe multiterminaux à partir d'une solution optimale au Problème 2. Celui-ci est du à Calinescu et al. [3].

Pour cela, introduisons quelques notations. Pour tout $1 \leq i \leq k$, notons $E_i = \{uv \in E \mid x_u^i \neq x_v^i\}$ et $W_i = \sum_{e\in E_i} w(e)d(e)$. Enfin, pour $\rho \in [0.1]$ et $1 \leq i \leq 1$, $B(s_i,\rho) = \{v \in V, x_v^i \geq \rho\}$. Notons si $\rho > 1/2$ alors pour $i \neq j$ $B(s_i,\rho) \cap B(s_j,\rho) = \emptyset$.

Algorithme 1 (Calinescu et al. [3])

1. Trouver une solution optimale x au Problème 2.
2. Modifier l'instance et la solution optimale suivant le Lemme 3.
3. Renuméroter de manière à ce que $W_k = \max\{W_i \mid 1 \leq i \leq k\}$.
4. Tirer ρ au hasard dans $[0,1]$ et choisir au hasard une permutation σ parmi les deux $(1,2,\ldots,k-1,k)$ et $(k-1,k-2,\ldots,1,k)$.
5. Pour $i = 1$ à $k-1$, $V_{\sigma(i)} := B(s_i,\rho) \setminus \bigcup_{j<i} V_{\sigma(j)}$.
6. $V_k := V \setminus \bigcup_{j<i} V_{\sigma(j)}$.
7. Retourner la coupe $(V_1,\ldots,V_k)$ (expurgée des sommets crée à l'étape 2).

Nous allons maintenant montrer que l'algorithme 1 renvoie une solution $(\frac{3}{2}-\frac{1}{k})$-approchée, c'est-à-dire que $\mathbf{E}(w(\Pi)) \leq (\frac{3}{2}-\frac{1}{k})e(\Pi_{opt})$ avec Π_{opt} une $(s_1,\ldots,s_k)$-coupe de poids minimun.

Lemme 4 *(i) Si $e \in E \setminus E_k$ alors* $\mathbf{Pr}(e \text{ est transverse}) \leq \frac{3}{2}d(e)$.

(ii) Si $e \in E_k$ alors $\mathbf{Pr}(e \text{ est transverse}) \leq d(e)$.

Preuve. (i) Posons $e-uv$. Soient i etv j les deux coordonnées pour lesquelles x_u et x_v diffèrent. Sans perte de généralité, on peut supposer $x_u^i < x_v^i$ et donc $x_u^j > x_v^j$. On peut également supposer $x_u^i < x_v^i$. Comme x_u et x_v ne diffèrent qu'en deux coordonnées exactement et la somme des coordonnées de ces vecteurs vaut 1 alors les segments $[x_u^i, x_v^i]$ et $[x_v^j, x_u^j]$ sont de même longueur $d(uv)$. En particulier, $x_u^j > x_v^j$.

Nous définissons $B = [x_v^j, x_u^j]$ et $A = [x_u^i, x_v^i]$ si $x_v^i < x_v^j$ et $A = [x_u^i, x_v^j]$ sinon.

Assertion 1 *Si $\rho \notin A \cup B$ alors u et v sont dans la même partie.*

Preuve.

□

Clairement, $\mathbf{Pr}(\rho \in A \cup B) = |A| + |B| \leq 2d(e)$. Mais cela ne permet de montrer que la fait que l'algorithme 1 donne une solution 2-approchée. Or nous voulons montrer qu'il produit une solution $(3/2-1/k)$-approchée.

Assertion 2 *Si $\rho \in A$ et $\sigma(j) < \sigma(i)$ alors u et v sont dans la même partie.*

Preuve.

□

Or $\mathbf{Pr}(\rho \in A \text{ et } \sigma(j) < \sigma(i)) = \mathbf{Pr}(\rho \in A) \times \mathbf{Pr}(\sigma(j) < \sigma(i)) = d(e)/2$ car les deux événements "$\rho \in A$" et "$\sigma(j) < \sigma(i)$" sont indépendants. Ainsi $\mathbf{Pr}(e \in E(\Pi)) \leq \mathbf{Pr}(\rho \in A \cup B) - \mathbf{Pr}(\rho \in A \text{ et } \sigma(j) < \sigma(i)) \leq \frac{3}{2}d(e)$. □

Pour finir,

$$\mathbf{E}(w(\Pi)) = \sum_{e \in E} w(e)\,\mathbf{Pr}(e \in E(\Pi)) \leq \sum_{e \in E_k} w(e)d(e) + \frac{3}{2}\sum_{e \in E \setminus E_k} w(e)d(e) \leq \left(\frac{3}{2}-\frac{1}{k}\right)\sum_{e \in E} w(e)d(e)$$

car $W_k = \max\{W_i \mid 1 \leq i \leq k\} \geq \frac{1}{k}\sum_{e \in E} w(e)d(e)$.

Le Problème 2 étant une relaxation du problème de la coupe multiterminaux, le minimum atteint par la solution optimale du Problème 2 est inférieur ou égal au poids de la coupe multiterminaux i.e. $\sum_{e \in E} w(e)d(e) \leq w(\Pi_{opt})$. Ainsi $\mathbf{E}(w(\Pi)) \leq \left(\frac{3}{2}-\frac{1}{k}\right) w(\Pi_{opt})$.

Il reste cependant à dérandomiser cet algorithme. Le lecteur intéressé est renvoyé vers **ref**.

2 Programmation semi-définie positive et application à la coupe maximale

Problème 5 (Coupe maximale) **Instance:** Un graphe $G = (V,E)$, une valuation des arêtes $w : E \to \mathbb{R}^+$.
Question: Trouver une bipartition (V_1, V_2) de V de poids maximun.

Ce problème est bien connu pour être NP-difficile. **ref**.

Le but de cette partie est de présenter l'algorithme polynomial donné par Goemans et Williamson [6] qui donne une solution 0.8785-approchée au probème de la coupe maximale. Cet algorithme est basé sur la programmation semi-définie positive que nous présentons brièvement.

2.1 Programmation semi-définie positive

$\varphi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ est une *forme bilinéaire* si

$$\begin{aligned} \varphi(x+\lambda.y,z) &= \varphi(x,z)+\lambda\varphi(y,z) \\ \varphi(x,y+\lambda.z) &= \varphi(x,y)+\lambda\varphi(x,z) \end{aligned}$$

$q : \mathbb{R}^n \to \mathbb{R}$ est une *forme quadratique* s'il existe une forme bilinéaire φ telle que $q(x) = \varphi(x,x)$.

Notons que si q est une forme quadratique, on peut facilement retrouver la forme bilinéaire associée: en effet, φ définie par $\varphi(x,y) = \frac{1}{2}(q(x+y) - q(x) - q(y))$ est une forme linéaire telle que $q(x) = \varphi(x,x)$.

Si ϕ est une forme bilinéaire, on peut l'écrire de façon matricielle: il existe une matrice $A \in \mathbb{R}^{n\times n}$ telle que

$$\forall (x,y) \in (\mathbb{R}^n)^2, \varphi(x,y) = x^T Ay.$$

De même, une forme quadratique peut s'écrire sous formae matricielle:

$$\forall x \in \mathbb{R}^n, q(x) = x^T Ac.$$

Une forme quadratique peut avoir plusieurs représentations matricielles. Par exemple,

$$\begin{pmatrix} x \\ y \end{pmatrix}^T \begin{pmatrix} 1 & 4 \\ 6 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = x^2 + 11xy + 2y^2 = \begin{pmatrix} x \\ y \end{pmatrix}^T \begin{pmatrix} 1 & 5 \\ 5 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

Cependant, une seule représentation est symétrique.

Une forme quadratique est *positive* si pour tout $x \in \mathbb{R}^n$, $q(x) \geq 0$. On note alors $q \succcurlyeq 0$. Si A est une représentation matricielle de q, on note également $A \succcurlyeq 0$. On note $\mathcal{U}_{PSD}$ l'ensemble des matrices symétriques A telle que $A \succcurlyeq 0$. Cet ensemble possède de nombreuses propriétés intéressantes.

Proposition 6 *(i) $\mathcal{U}_{PSD}$, i.e. si $A_1, A_2 \in \mathcal{U}_{PSD}$ et $\lambda \in [0,1]$ alors $\lambda A_1 + (1-\lambda)A_2 \in \mathcal{U}_{PSD}$.*

(ii) $\mathcal{U}_{PSD}$ est un cône, i.e. $\forall A \in \mathcal{U}_{PSD}, \lambda \in \mathbb{R}, \lambda A \in \mathcal{U}_{PSD}$.

(iii) Tester si $A \in \mathcal{U}_{PSD}$ est polynomial.

Un *programme semi-défini positif* est un problème de la forme suivante: Etant donné un ensemble de matrices $A^i \in \mathbb{R}^{n\times n}$, $1 \leq i \leq m$ et $C \in \mathbb{R}^{n\times n}$ et des réels b_i, $1 \leq i \leq m$, trouver $X \in \mathcal{U}_{PSD}$ telle que pour tout $1 \leq i \leq m$ $\text{trace}(A^i X) = b_i$ qui maximise $\text{trace}(CX)$.

De même qu'un programmes linéaire, un programme semi-défini positif possède un dual dont le résultat est en général le même. Ce dual consiste à trouver des réels $y_1, \ldots, y_m$ tels que $\sum_{i=1}^m y_i A^i - C \in \mathcal{U}_{PSD}$ pour lesquels $\sum_{i=1}^m b_i y_i$ est minimum.

Théorème 7 *Résoudre un problème semi-défini positif se fait en temps polynomial.* **ref**

2.2 Coupe maximale par la programmation semi-définie positive

Soit $G = (V,E)$ un graphe à n sommets et w une valuation des arêtes de G. On étend w à tout $V \times V$ en posant $w(u,v) = w(e)$ si $uv \in E$ et $w(u,v) = 0$ sinon.

Le problème de la coupe maximale peut-être reformulé de la manière suivante:
Trouver les $y_v \in \{-1,1\}$ pour $v \in V$ qui maximise $\sum_{e\in E} w(e) = \frac{1}{2}\sum w(u,v)(1-y_u y_v)$. est maximale. La coupe est alors donnée par $V_1 = \{v \in V \mid y_v = 1\}$ et $V_2 = \{v \in V \mid y_v = -1\}$.

Ce n'est pas un programme semi-défini positif mais on peut le relâcher pour que cela le soit. Pour cela au lieu de prendre des réelsl y_v dans $\{-1,1\}$, on prend des vecteurs Y_v dans $\{x \in \mathbb{R}^n \mid ||x|| = 1\}$.

Considérons alors la matrice X définie par $X_{u,v} = Y_u.Y_v$.

Proposition 8 *(i)* $X_{u,u} = 1$.

(ii) $X \in \mathcal{U}_{PSD}$.

Ainsi trouver les Y_v dans $\{x \in \mathbb{R}^n \mid ||x|| = 1\}$ qui maximise $\sum_{e \in E} w(e) = \frac{1}{2}\sum w(u,v)(1 - Y_u Y_v)$ est équivalent à trouver $X \in \mathcal{U}_{PSD}$ qui minimise $\sum w(u,v)X_{u,v}$ sous la contrainte $X_{u,u} = 1$ pour tout $u \in V$. Ceci est un problème semi-défini positif avec $C = (w_u, v)$ et A^i la matrice tel que $A^i_{i,i} = 1$ et $A^i_{j,k} = 0$ si $(j,k) \neq (i,i)$.

L'algorithme de Goemans et Williamson pour approximer la coupe maximal est alors le suivant.

Algorithme 2 (Goemans et Williamson [6])

1. Résoudre le problème semi-défini positif ci-dessus.
2. Tirer un vecteur $Z \in \mathbb{R}^n$ au hasard.
3. Pour tout sommet $v \in V$, si $Y_v.Z \geq 0$ alors $y_v = 1$ et $y_v = -1$ sinon.
4. Rendre (V_1, V_2) avec $V_1 = \{v \in V \mid y_v = 1\}$ et $V_2 = \{v \in V \mid y_v = -1\}$.

Estimons maintenant la qualité de la solution rendue par cet algorithme. Une arête uv est transverse si C est dans le secteur angulaire formé par Y_u et Y_v. Ainsi $\mathbf{Pr}(uv \text{ transverse}) = \frac{\text{Arccos}(Y_u.Y_v)}{\Pi}$.

Soit ALG le poids de la coupe rendu par l'Algorithme 2, OPT le poids maximal d'une coupe et SDP la solution du problème semi-défini positif obtenu en relaxant. Alors $OPT \leq SDP$. Ainsi

$$\begin{aligned}
\frac{\mathbf{E}(ALG)}{OPT} &\geq \frac{\mathbf{E}(ALG)}{SDP} \\
&\geq \frac{\sum w_{u,v} \frac{1}{\Pi} \text{Arccos}(Y_u.Y_v)}{\sum w_{u,v} \frac{1}{2}(1 - Y_u.Y_v)} \\
&\geq \min\left\{ \frac{2}{\pi} \times \frac{\text{Arccos}(t)}{1-t}, t \in [0,1] \right\} \\
&\geq 0,8785.
\end{aligned}$$

Là aussi il faut dérandomiser.

3 Méthode du ratio local

Nous allons illustrer cette méthode par le problème des t-intervalles disjoints.
Un *t-intervalles* est l'union de t-intervalles.

Problème 9 (t-intervalles disjoints)
Instance: Un ensemble $\mathcal{I}$ de t-intervalles.
Question: Trouver un sous-ensemble de t-intervalles deux à deux disjoints de cardinalité maximum.

Remarque 10 Ce problème revient à trouver le stable maximum dans le graphe d'intersection des t-intervalles.

Pour $t = 1$, ce problème est facile et se résoud par l'algorithme glouton. On prend l'intervalle dont l'extrémité droite est la plus à gauche. On note tous les intervalles qui l'intersectent (et qui s'intersectent deux à deux) et on recommence. Pour $t \geq 2$, le problème est APX-hard [2].

Le Problème 9 peut se formuler sous la forme d'un programme linéaire en nombres entiers.

$$\begin{array}{rll} \text{Maximiser} & \sum_{I\in\mathfrak{I}} y_I & \\ \text{Sous les contraintes :} & \sum_{I\ni z} y_I \leq 1 & \forall z \in \mathbb{R} \\ & y_I \in \{0,1\} & \end{array}$$

Un tel problème se relaxe classiquement en une version fractionnaire en remplaçant la condition non-linéaire $y_I \in \{0,1\}$ par $y_I \geq 0$. Notons qu'il est possible d'obtenir un programme linéaire équivalent avec un nombre polynomial de contraintes en ne prenant que celles dont le z correspond à l'extrémité droite d'un segment d'un t-intervalle. Ce nouveau programme linéaire peut alors être résolu en temps polynomial.

Problème 11 (t-intervalle fractionnaire)

$$\begin{array}{rll} \text{Maximiser} & \sum_{I\in\mathfrak{I}} y_I & \\ \text{Sous les contraintes :} & \sum_{I\ni z} y_I \leq 1 & \forall z \in \mathbb{R} \\ & y_I \geq 0 & \end{array}$$

Le dual de ce problème est le suivant :

Problème 12 (t-intervalle fractionnaire dual)

$$\begin{array}{rll} \text{Minimiser} & \sum_{z\in\mathbb{R}} x_z & \\ \text{Sous les contraintes :} & \sum_{z\in I} x_z \geq 1 & \forall I \in \mathfrak{I} \\ & x_z \geq 0 & \end{array}$$

Pour un t-intervalle $I \in \mathfrak{I}$, notons $S(I) = \{J \in \mathfrak{I} \mid I \cap J \neq \emptyset\}$, c'est-à-dire, l'ensemble des t-intervalles qui ne peuvent pas se trouver avec I dans une solution admissible. Dans le cas $t = 1$, l'algorithme glouton est basé sur la propriété suivante.

Proposition 13 *Il existe un intervalle $I^* \in \mathfrak{I}$ tel que $\bigcap_{I\in S(I^*)} I \neq \emptyset$, donc il existe $z^* \in R$ tel que $z^* \in \bigcap_{I\in S(I^*)} I$.*

Cette proposition permet d'obtenir un algorithme primal-dual. Elle nous dit qu'il existe un élément $\mathfrak{I}^*$ du problème primal tel que tous les éléments qui ne peuvent pas se trouver avec lui dans la solution primale peuvent être intersectés par un élément z^* du problème dual. Cela permet de construire une solution primale et une solution duale en ajoutant simultanément $\mathfrak{I}^*$ dans la solution du primal et z^* dans la solution du dual. Les éléments qui contiennent z^* sont supprimés et le processus recommence tant qu'il reste des t-intervalles non intersectés par la solution du dual. Comme, à chaque étape de l'algorithme, nous faisons grandir de la même valeur la solution du primal et du dual, nous obtenons, à la fin du processus, une solution du primal et une solution du dual ayant la même valeur. Cela garantit l'optimalité des solutions.

Pour $t \geq 2$, on pourait espérer avoir une approximation en utilisant là aussi une approche primal-dual mais en faisant augmenter la solution du primal et du dual de valeurs différentes. Si on pouvait montrer qu'on peut prendre k éléments dans le primal qui nécessitent k' éléments dans le dual alors on obtiendrait une k'/k-approximation.

Malheureusement, une telle approche n'est pas possible pour le problème des t-intervalles disjoints.

Proposition 14 *Pour tout $n \geq 2$, il existe un ensemble $\mathfrak{I}$ de 2-intervalles tel que, pour tout $I \in \mathfrak{I}$, $S(I)$ contient un ensemble de $\Omega(\sqrt{n})$ 2-intervalles disjoints.*

Preuve. Voir la preuve dans [2]. □

Ainsi on ne peut pas utiliser la méthode primal-dual telle quelle. On va alors utiliser la méthode du ratio locale. Cela consiste à calculer une solution optimale du Problème 11 qui en toute généralité n'est pas entière et ensuite d'utiliser une propriété locale pour l'arrondir en une solution entière.

Pour $t \geq 2$ nous allons donner un algorithme $2t$-approché. Celui-ci est basé sur la proposition suivante.

Proposition 15 *Soit y une solution du Problème 11. Alors il existe $I^* \in \mathfrak{I}$ tel que* $\sum_{I \in S(I^*)} y_I \leq 2t$.

Preuve. Pour tout intervalle $i \in I$, notons $R(i)$ l'ensemble des t-intervalles de $\mathfrak{I}$ contenant l'extrémité droite de i. Comme deux t-intevalles s'intersectent si et seulement si l'un des deux à une extrémité droite dans l'autre. Alors

$$\sum_{I \in \mathfrak{I}} \sum_{J \in S(I)} y_I y_J \leq 2 \sum_{I \in \mathfrak{I}} \sum_{i \in I} \sum_{J \in R(i)} y_I y_J.$$

Or $\sum_{i \in I} \sum_{J \in R(i)} y_J \leq t$ car il y a au plus t intervalles i dans I et $\sum_{J \in R(i)} y_J \leq 1$. Ainsi

$$\sum_{I \in \mathfrak{I}} \sum_{J \in S(I)} y_I y_J \leq \sum_{I \in \mathfrak{I}} 2t y_I.$$

Il existe donc $I \in \mathfrak{I}$ tel que $\sum_{J \in S(I)} y_I y_J \leq 2t y_I$ soit $\sum_{J \in S(I)} y_J \leq 2t$. □

L'algorithme est alors le suivant:

Algorithme 3 ([2])

1. Trouver une solution optimale y du Problème 11.
2. Trouver $I^* \in \mathfrak{I}$ tel que $\sum_{J \in S(I^*)} y_J \leq 2t$.
3. $y_{I^*} := 1$ et $y_J := 0$ pout tout $J \in S(I^*) \setminus I^*$; $\mathfrak{I} := \mathfrak{I} \setminus S(I^*)$.
4. Si $\mathfrak{I}$ est non vide alors aller à l'étape 2.

Remarque 16 La méthode peut se généraliser à une version pondérée dans laquelle les t-intervalles ont un poids et où on cherche un ensemble de t-intervalles de poids maximum.

4 Programmation linéaire et algorithmes paramétrés

Dans cette partie, nous allons voir une application de la programmation linéaire à la complexité paramétrée. Nous renvoyons le lecteur aux notes de la JCALM [7] sur ce domaine pour les définitions et autres omises ici.

Une *couverture* d'un graphe $G = (V, E)$ est un ensemble de sommets $S \subset V$ tel que toute arête de E a au moins une de ses extrémités dans S. Une couverture est dite *minimum* si elle est de cardinalité minimum. La cardinalité d'une couverture minimum de G est dénotée $\nu(G)$.

Problème 17 (Couverture minimum (paramétré))
Instance: Un graphe G et un entier k.
Paramètre: k.
Question: G a-t-il une couverture de taille au plus k ? i.e. $\nu(G) \leq k$?

Nous avons vu dans la JCALM sur les algorithmes paramétrés que le problème de la couverture minimum était FPT. Il admet donc un noyau. Nous avons par ailleurs vu (Lemme 14 de [7]) que le problème de la couverture minimum a un noyau de taille quadratique: k^2+k pour être précis. Le but de cette partie est de montrer qu'il a un noyau linéaire.

Théorème 18 *Le problème de la couverture minimum a un noyau de taille* $2k$.

Le problème de la couverture minimum peut être relâché en programmation linéaire.

Problème 19 (couverture minimum fractionnaire)

$$\begin{array}{rll} \text{Minimiser} & \sum_{v\in V} x_v & \\ \text{Sous les contraintes :} & x_u + x_v \geq 1 & \forall uv \in E \\ & q \leq x_v \geq 1 & \forall v \in V \end{array}$$

Théorème 20 *Le problème de la couverture minimum fractionnaire (Problème 19) a une solution optimale* $1/2$*-entière, i.e. telle que* $x_v \in \{0,1/2,1\}$ *pout tout* $v \in V$.

Preuve. Soit x une solution optimale ayant le plus de coordonnées dans $\{0,1/2,1\}$.

Supposons que x n'ait pas toutes ses coordonnées dans $\{0,1/2,1\}$. Posons $\varepsilon = \min\{x_v, |x_v - \frac{1}{2}|, 1-x_v \mid v \in V \text{ et } x_v \notin \{0,1/2,1\}\}$.

Considérons x' et x'' définies comme suit:

$$x'_v = \leq \begin{cases} x_v - \varepsilon, & \text{si } 0 < x_v < \frac{1}{2}, \\ x_v + \varepsilon, & \text{si } \frac{1}{2} < x_v < 1, \\ x_v, & \text{sinon.} \end{cases} \quad \text{et } x''_v = \leq \begin{cases} x_v + \varepsilon, & \text{si } 0 < x_v < \frac{1}{2}, \\ x_v - \varepsilon, & \text{si } \frac{1}{2} < x_v < 1, \\ x_v, & \text{sinon.} \end{cases}$$

Ce sont deux solutions admissibles pour le Problème 19. De plus, $\sum_{v\in V} x_v = \frac{1}{2}\left(\sum_{v\in V} x'_v + \sum_{v\in V} x''_v\right)$. Ainsi x' et x'' sont elles aussi optimales et par le choix de ε une de ces deux solutions a plus de coordonnées dans $\{0,1/2,1\}$ que x, une contradiction. □

Soit x une solution optimale $1/2$-entière du Problème 19. Pour $t \in \{0,1/2,1\}$, posons $V_t = \{v \in V \mid x_v = t\}$, $G_t = G\langle V_t\rangle$.

Observation 1 *Si* S *est une couverture de* G *alors* S_t *est une couverture de* G_t.

Observation 2 *Si* S' *est une couverture de* $G_{1/2}$ *alors* $S' \cup V_1$ *est une couverture de* G.

Preuve. Soit uv une arête de G. Alors soit u et v sont dans $V_{1/2}$ et alors uv est couverte par S' ou alors u ou v est dans V_1 et uv est couverte par ce sommet. □

Observation 3 $\nu(G_{1/2}) \geq \frac{1}{2}|V_{1/2}|$.

Preuve. $\nu(G_{1/2}) + |V_1| \geq \nu(G) \geq \sum_{v\in V} x_v = \frac{1}{2}|V_{1/2}| + |V_1|$. □

Théorème 21 (Nemhauser et Trotter [9]) *Il existe une couverture minimum* S *de* G *tel que:*

(a) $V_0 \cap S = \emptyset$;

(b) $V_1 \subseteq S$.

Preuve. La preuve suivante est due à Khuller [8]. Montrons tout d'abord que (a) implique (b).
Supposons qu'il existe $v \in V_1$ qui n'appartienne pas à S.

- Si v n'a aucun voisin dans V_0 alors on peut diminuer le poids x_v en v pour obtenir une solution fractionnaire de moindre poids. Cela contredit l'optimalité de x.
- Si v a un voisin w dans V_0 alors vw n'est pas couverte car $S \cap V_0 \neq \emptyset$.

Montrons maintenant (a).
Soit S une couverture minimum.

Supposons que $S \cap V_0 \neq \emptyset$. avec le nombre minimum de sommet dans V_0. Observons que V_0 est un stable car x est une couverture fractionnaire et qu'il n'y a pas d'arêtes entre $V_0 \setminus S$ et $V_1 \setminus S$.

Supposons que $|V_0 \cap S| < |V_1 \setminus S|$. Alors on peut augmenter x_v de ε pour $v \in V_0 \cap S$ et diminuer x_v de ε pour tout $v \in V_1 \setminus S$. On obtient alors une couverture fractionnaire de G de poids strictement plus petit que celui de x, une contradiction.

On a donc $|V_0 \cap S| \geq |V_1 \setminus S|$ alors $(S \setminus V_0) \cup V_1$ est une couverture de G avec au plus autant de sommet que S et aucun sommet dans V_0. □

L'Observation 3 et le Théorème 21 implique immédiatement que

Corollaire 22 $\nu(G_{1/2}) + |V_1| = \nu(G)$.

Nous pouvons maintenant prouver le Théorème 18.
L'algorithme de réduction à un noyau est le suivant.

Algorithme 4

1. Trouver une solution optimale x du Problème 19.
2. Si le poids de x est supérieur à k, renvoyer une instance "non".
3. Si $|V_1| = k$ et $V_{1/2} = \emptyset$, renvoyer instance "oui"
4. Sinon renvoyer $(G_{1/2}, k - |V_1|)$.

Cela donne bien un noyau de taille $2k$ car lorsque l'on renvoie $(G_{1/2}, k - |V_1|)$, nous avons $|V_{1/2}| \leq 2(k - |V_1|)$ par l'Observation 3.

References

[1] R. Bar-Yehuda, K. Bendel et D. Ravitz. Local Ratio : A unified framework for approximation algorithm. *ACM surveys* 36:422-463, 2004.

[2] R. Bar-Yehuda, M. M. Halldorsson, J. Naor, H. Shachnai and I. Shapira, Scheduling Split Intervals, 13th ACM-SIAM Symposium on Discrete Algorithms (SODA), January 6-8, 2002.

[3] G. Calinescu, H. Karloff et Y. Rabani. An improved approximation algorithm for multiway cut. *Journal of Computer and System Sciences* 60:564-574, 2000.

[4] E. Dahlhaus, D.S. Johnson, C. H. Papadimitriou, P. D. Syemour et M. Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.* 23:864–894, 1994.

[5] J. Flum et M. Grohe. Kernelization and linear programming techniques. In *Parameterized Complexity Theory*, Chapter 9, pp. 207-231, 2006.

[6] M. X. Goemans et D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. of the ACM* 42(6):1115–1145, 1995.

[7] F. Havet et C. Paul. Introduction à la complexité paramétrée. *Notes des JCALM du 6 juin 2008.*

[8] S. Khuller. The Vertex Cover problem. *ACM SIGACT News* 33(2):31–33, 2002.

[9] G. L. Nemhauser and L. E. Trotter. Vertex packings: Structural properties and algorithms. *Math. Program.* 8:232–248, 1975.